Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

The edge of optimization in large-scale vehicle routing for paratransit

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, dbertsim@mit.edu

Julia Yan
Operations Research Center, Massachusetts Institute of Technology, jyyan@mit.edu

The Americans with Disabilities Act of 1990 mandates that accessible transportation be provided to those who cannot use the regular transit system due to disability. Paratransit agencies operate fleets of vehicles to fulfill daily requests for transportation in shared rides, which are typically collected one to seven days ahead of time. Although paratransit is an essential safety net, it is also expensive to operate and requires large government subsidies. These financial difficulties, combined with significant improvements in mixed integer optimization (MIO) solvers in recent years, have led to interest in developing large-scale optimization algorithms for paratransit. The classical problem in this field is called the *dial-a-ride problem*. In this paper, we provide a cluster-then-route approach to servicing paratransit requests subject to labor constraints. We demonstrate the tractability and efficacy of our algorithms for day-ahead planning as well as day-of reaction to uncertainty. Our case study is based on real data from Boston, MA ranging from 3,000 to 7,000 requests per day. On typical weekday instances, our algorithms improve upon an insertion heuristic by over 30% and against a competitive clustering baseline by over 10% in productivity (requests served per hour).

Key words: vehicle routing, dial-a-ride, integer optimization, paratransit
History:

1. Introduction

The fixed-route public transit system, which includes buses and subways, can pose significant obstacles to people with disabilities. For example, boarding a bus or accessing a subway platform may require climbing stairs that are challenging or impossible for elderly and wheelchair-bound commuters. This problem is particularly widespread in New York City, where only a quarter of subway stations are wheelchair-accessible, and where a recent plan to upgrade these stations was projected to come at the enormous cost of $78 million per station (Fitzsimmons and Liebson 2019). Even in cities with full accessibility, accommodating passengers with disabilities on fixed routes

can be challenging due to the extra time often required to assist with boarding and alighting the vehicles.

The Americans with Disabilities Act (ADA) of 1990 mandates that transportation options be provided for those who cannot take the existing fixed-route public transit system due to disability. This service typically involves the operation of a fleet of vehicles that take passengers from door-to-door or curb-to-curb in shared rides. Paratransit is intended to be comparable in service quality to the fixed-route system in service area, operating hours, fares, and travel times: the ADA stipulates that paratransit must operate within three-quarters of a mile from any fixed-route station, have the same operating hours, be no more than twice as expensive, and have reasonable travel times.

One result of these federal standards is that paratransit is notoriously expensive. According to the Citizens Budget Commission (2016), the Access-a-Ride program in New York City incurred $472 million of expenses in 2015 over the course of over six million rides. Only 4% percent of these expenses were covered by revenues from the service, with the remainder covered by taxes (about 13%) and subsidies (about 83%). In Boston in 2015, The RIDE serviced two million rides and cost over $100 million, with only 6% of this cost covered by revenue from the service (Massachusetts Bay Transportation Authority 2015). Furthermore, paratransit expenses are likely to increase dramatically as more become eligible for the service due to the dual threats of urbanization and the aging of the American population; the number of Americans 65 years or older is projected to rise from 46 million in 2015 to more than 98 million in 2060 (Mather et al. 2015).

High financial costs have put pressure on paratransit agencies to improve their operations. One cost-saving initiative focuses on algorithmic improvements to the systems that generate routes for drivers. The *dial-a-ride problem* (DARP) is a classical problem in this area, and models the problem of routing a fleet of vehicles to fulfill a set of transportation requests at minimum cost. In paratransit, each request for transportation comprises the desired origin and destination locations, and may also include time windows on departure arrival times as well as characteristics including the number of passengers and whether any passenger requires extra accommodation, such as for a wheelchair or service animal. The problem is further complicated by operating constraints, which generally include capacity constraints on the vehicles, labor constraints on the drivers, as well as service guarantees for passengers such as maximum ride time. Beyond paratransit, the DARP models decisions faced by ridesharing services such as Uber Pool, Lyft Line, or Via, all of which aim to connect multiple requests in a single vehicle to reduce costs.

Another idea generating interest among paratransit agencies is the option of outsourcing rides to Transit Network Companies (TNCs) such as Uber or Lyft, which can typically fulfill requests at lower cost. Agencies exploring TNC partnerships or partnerships with the local taxi industry include those of Boston, Massachusetts; San Bernardino, California; and Denton County, Texas

(American Public Transit Association 2019). Other agencies in Chicago, Illinois and Denver, Colorado have partnered with the local taxi industry, and New York City previously had such a partnership from 2010 to 2015 (Citizens Budget Commission 2016). Outsourcing allows paratransit agencies to avoid high costs that may be incurred by single requests that cannot fit easily into a driver's itinerary. Despite the lower costs associated with TNCs, it is unlikely that private ridesharing will completely replace paratransit: first, because of safety issues and discrimination that have been alleged by disabled customers (Sullivan 2014, Binette and Vasold 2018); and second, because Uber itself has contracted out some of its wheelchair-accessible service to paratransit companies (Siddiqui 2018). As such, it will continue to be important to study the paratransit operating model, while also identifying a limited number of requests that can be productively outsourced.

To date, there has been a gap between cutting-edge research and real-world impact. Some approaches may provide problem insight but are based on simplified, unrealistic operating strategies. Others may be complex black boxes that are challenging to implement and prevent buy-in from practitioners. Finally, a persistent problem is one of scalability: providing high-quality decisions at the scale of a major metropolitan area is an enormous challenge and can require heavy computation time. Because all paratransit transportation requests are collected a day in advance, there may appear to be several hours available for computation of a solution. However, in practice, substantial time must be allotted to actually implement the solution, and repeated re-calculation may be necessary if constraints were omitted in early runs or in response to unexpected events such as driver breakdowns or passenger cancellations.

We aim to bridge the gap between research and impact by providing intuitive models with straightforward implementation and practical running times on real-world data from Boston, MA. Our case studies comprise problem sizes of 3,000 to 7,000 requests, which to our knowledge is the largest DARP application examined in the literature. Algorithmic characteristics such as parallelizability, fast generation of an initial solution, and iterative improvement all contribute to fast running times and allow for a balance between performance and tractability.

Our approach falls into a stream of work broadly categorized as *cluster-then-route*, where transportation requests are grouped together into clusters, then vehicles are routed within each cluster, and then finally vehicles are routed between clusters. In this framework, we make the following contributions:

1. A clustering routine that takes a divide-and-conquer-like approach to quickly find high-quality solutions at scale, then iteratively improves the solutions. Requests are first grouped into small clusters on which it is manageable to optimally solve the DARP. Subsequent iterations combine clusters together, using fast optimization to guide the algorithm towards profitable combinations, and re-solve the DARP on these combined clusters in order to continually improve solution quality.

2. An integer optimization model that generates drivers' itineraries from the output of the previous step and also decides which requests to outsource to TNCs, subject to a variety of labor constraints. The formulation accounts for a variety of time windows, capacity constraints, heterogeneous vehicle types, and shift start and end times.
3. An end-to-end perspective that touches various aspects of the planning process, from tactical day-ahead routing decisions, to day-of reaction to uncertainty. A critical aspect to applicability in these domains is tractability. Our algorithms run on large-scale problem sizes of several thousand requests in under twenty minutes when implemented in serial, and can be sped up even further with parallelization.
4. Our overall algorithm improves upon an insertion heuristic from Marković et al. (2015) by over 30% and a clustering method from Santi et al. (2014) by over 10% in productivity (requests served per hour) on typical weekday instances.

The remainder of the paper is outlined as follows. Section 2 reviews the literature on various approaches to the DARP as well as related problems. Section 3 discusses our cluster-then-route methodology for solving the DARP at scale. Section 4 discusses the paratransit operating model in Boston, with resulting algorithmic considerations that are required for practical use. Section 5 covers computational results on real data from Boston, which has approximately 7,000 passengers on weekdays and 3,000 on weekends. We conclude in Section 6.

2. Literature Review

The DARP is well-studied in the literature. For comprehensive reviews, see Cordeau and Laporte (2007) and Baldacci et al. (2012). The focus of this paper is on the *static* problem, in which all requests are known ahead of time. The *dynamic* variant, where batches of requests are revealed at small time intervals, is covered in a review by Agatz et al. (2012).

A common approach to solving large-scale problems is to use heuristics or metaheuristics such as tabu search (Cordeau and Laporte 2003), insertion heuristics (Diana and Dessouky 2004, Marković et al. 2015), or large neighborhood search (Jain and Van Hentenryck 2011). Although these approaches can scale to hundreds or thousands of requests, they may also require tuning a large number of parameters that can be sensitive to input data, which is not ideal for widespread implementation.

Exact integer optimization methods have been tested up to hundreds of requests. Savelsbergh and Sol (1998) use column generation to solve pickup and delivery problems of up to 300 requests. Similarly, Baldacci et al. (2004) derive lower bounds based on column generation and solve problems with up to 250 requests. Ropke et al. (2007) introduce several types of valid inequalities that strengthen the formulations, and solve problems with up to 96 requests.

Bridging these two frameworks is a category called *cluster-then-route*, which adopts the strategy of breaking down large-scale problems into a sequence of smaller optimization problems that are individually more tractable. This approach is necessitated by the fact that while problems on even a few hundreds of requests can be challenging to solve, real-world applications can easily involve several thousands of requests. The approach first groups transportation requests into clusters, generates vehicle routes within each cluster, and then connects the cluster routes. The first step of constructing the clusters is of particular importance, as the effects of poor clustering will propagate downstream in subsequent steps of the algorithm.

A simple approach to clustering cited in Savelsbergh and Sol (1998) as one often used in practice is to group the requests based on the proximity of origins and destinations in space and time. Early works such as Ioachim et al. (1995) and Borndörfer et al. (1999) focus especially on the first clustering step with more sophisticated approaches. Ioachim et al. (1995) used column generation to generate clusters for a problem of 2,545 requests, although the approach did not scale to the full set of requests and the problem was split into smaller batches of about 150 requests each. Borndörfer et al. (1999) used complete enumeration and a set cover formulation to generate clusters on problem sizes ranging from 1,000 to 1,500 requests. The scalability of complete enumeration here relied critically on the fact that the problem parameters were restricted enough that it was often not possible to service multiple requests at once; for example, the vehicle could be diverted at most fifteen minutes to pick up another passenger. In more flexible settings with wider time windows, such an approach would not be scalable.

Nevertheless, limited enumeration can still be a powerful tool in large-scale problems. Ioachim et al. (1995) and, more recently, Santi et al. (2014) both relied on limited enumeration to construct their solutions, using the fact that it is easy to evaluate the four possible ways to fulfill a pair of requests in a shared trip. Ioachim et al. (1995) use this enumeration to reduce the search space in the subproblem of the column generation algorithm. Meanwhile, Santi et al. (2014) used a graph matching formulation to construct a list of pairs of requests to fulfill together, an approach that provides a high-quality solution with only moderate computational expense even at the scale of the New York City taxicab dataset.

Much of the aforementioned literature has focused on the DARP without fully addressing auxiliary operating constraints, due to the difficulty of solving even the base problem. However, such constraints are crucial to real-world application. A variety of approaches have also been proposed for the operating constraints that transit agencies must obey. Driver shifts are commonly restricted in length, which is addressed in the exact optimization approach of Parragh et al. (2012) and the insertion heuristic of Marković et al. (2015). Passengers may also be obliged to make transfers, which is addressed using exact optimization by Cortés et al. (2010), evolutionary algorithms by Herbawi and Weber (2011), and large neighborhood search by Masson et al. (2014).

3. Methods

We will take a cluster-then-route approach to large-scale DARPs in paratransit, which breaks the problem down into a sequence of smaller optimization problems that are individually more tractable. The approach uses the following definitions:

- Every *request* for transportation specifies a pickup location, a dropoff location, time windows on allowed pickup and dropoff times, and characteristics such as number of passengers and whether any passengers require special accommodations. A request might either be a "pickup" or an "appointment"; the passenger names a desired pickup time for pickup requests, and a desired dropoff time for appointment requests. Time windows are formed around the time named by the passenger, and can be computed for the other endpoint of the request depending on travel time and the maximum allowed time of a passenger in the vehicle.
- A *trip* refers to a feasible sequence of pickups and dropoffs that will fulfill a set of requests such that the vehicle begins and ends empty but is occupied throughout. Feasible means that capacity constraints, time windows, and maximum ride time constraints are respected. For example, if i and j are requests, "pick up i, pick up j, drop off j, drop off i" represents a trip, assuming that it is feasible.
- An *itinerary* refers to a sequence of trips that a driver will follow over the course of the workday, which must also satisfy the driver's labor constraints. It should also begin and end at a garage.

In our approach, the first step is to partition the requests for transportation into a set of clusters, and then solve the DARP on the clusters to produce a set of trips. The second step is to connect the trips into driver itineraries, which is done by solving a problem that we call the Trip Connection Problem (TCP). The following section provides details of our approach and is outlined as follows.

- In Section 3.1, we summarize input parameters for our optimization problems, the DARP and the TCP.
- In Section 3.2, we formulate the DARP and describe our procedure to partition the requests into clusters and generate trips.
- In Section 3.3, we write the formulation of the TCP, which connects the trips in the previous step into driver itineraries.

3.1. Problem Statement

Descriptions of the input parameters for the DARP and TCP are given in Table 1. Both optimization problems seek to provide a transportation service at lowest cost; the difference is that the DARP starts from a set of individual requests that can be clustered together into the same vehicle, and the TCP starts with a set of trips that are fulfilled individually.

For the DARP, there are N *requests*, each of which is made of a pickup location, a pickup time window, a dropoff location, a dropoff time window, and the number of seats needed in the vehicle. These N requests induce a directed graph $\mathcal{G}(\mathcal{V}^{\text{DARP}}, \mathcal{E}^{\text{DARP}})$. The node set $\mathcal{V}^{\text{DARP}}$ contains nodes $\{0, 1, \ldots, 2N, 2N+1\}$, where 0 and $2N+1$ are the source and sink nodes, $\mathcal{P}^{\text{DARP}} = \{1, \ldots, N\}$ are the pickup nodes, and $\mathcal{D}^{\text{DARP}} = \{N+1, \ldots, 2N\}$ are the dropoff nodes. Each pickup node $i \in \mathcal{P}^{\text{DARP}}$ is associated with a corresponding node $i + N \in \mathcal{D}^{\text{DARP}}$. Each node $i \in \mathcal{V}^{\text{DARP}}$ is also associated with a service duration b_i, which in the case of paratransit could be load times at pickup nodes and unload times at dropoff nodes; a time window $[\ell_i, u_i]$ representing a range of possible arrival times; and a load d_i that represents the number of passengers for that request. Passengers are guaranteed to be diverted no more than W minutes. Finally, vehicle capacities are set to Q. In paratransit, as in the case of other logistics applications, it is common to have multiple types of passengers and vehicles; for example, sedans cannot accommodate wheelchair-bound passengers, so vans must fulfill those requests. Unless otherwise noted, we omit this distinction for ease of notation in writing out the formulation, but the generalization is straightforward.

Table 1 Summary of the main notation.

Symbol	Description
N	number of individual requests for door-to-door transit
$\mathcal{P}^{\text{DARP}}$	request pickup nodes, indexed as $1, \ldots, N$
$\mathcal{D}^{\text{DARP}}$	request dropoff nodes, indexed as $N+1, \ldots, 2N$
$\mathcal{V}^{\text{DARP}}$	all nodes for the DARP, including pickups $\mathcal{P}^{\text{DARP}}$, dropoffs $\mathcal{D}^{\text{DARP}}$, the source (index 0), and the sink (index $2N+1$)
$\mathcal{E}^{\text{DARP}}$	all valid edges between nodes in $\mathcal{V}^{\text{DARP}}$, including edges from the source and to the sink
A	number of trips made from clustering trips together
$\mathcal{P}^{\text{TCP}}$	first (pickup) node in each trip, indexed as $1, \ldots, A$
$\mathcal{D}^{\text{TCP}}$	last (dropoff) node in each trip, indexed as $A+1, \ldots, 2A$
$\mathcal{V}^{\text{TCP}}$	all nodes for the shift routing problem, including first pickups $\mathcal{P}^{\text{TCP}}$, last dropoffs $\mathcal{D}^{\text{TCP}}$, the source (index 0), and the sink (index $2A+1$)
$\mathcal{E}^{\text{TCP}}$	all valid edges between nodes in $\mathcal{V}^{\text{TCP}}$. Each pickup node only has one outgoing edge, which is to its corresponding dropoff node.
$c_{i,j}$	travel time between nodes i and j
b_i	service duration at node i
W	maximum time a passenger can be diverted
$[\ell_i, u_i]$	time window at node i
d_i	load at node i (positive at pickups and negative at dropoffs)
Q	vehicle capacity

3.2. Clustering Requests

We begin by providing a formulation for the DARP, which also appeared as PDPTW1 in Ropke et al. (2007). The decision variable $x_{i,j}$ for $(i,j) \in \mathcal{E}^{\text{DARP}}$ represents whether some vehicle travels

from node i to node j; the decision variable t_i for $i \in \mathcal{V}^{\text{DARP}}$ represents the arrival time at node i; and the decision variable q_i for $i \in \mathcal{V}^{\text{DARP}}$ represents the load of the vehicle as it arrives at node i. Then, trips that satisfy time window and capacity constraints at minimum cost can be found by solving the following optimization problem:

$$
\begin{aligned}
\text{(DARP)} \quad \min_{\mathbf{x},\mathbf{t},\mathbf{q}} \quad & \sum_{(i,j)\in\mathcal{E}^{\text{DARP}}} c_{i,j}x_{i,j} && && \text{(1a)}\\
\text{s.t.} \quad & \sum_{j\in\text{Out}(i)} x_{i,j} = 1 && \forall i \in \mathcal{V}^{\text{DARP}}, && \text{(1b)}\\
& \sum_{i\in\text{In}(j)} x_{i,j} = 1 && \forall j \in \mathcal{V}^{\text{DARP}}, && \text{(1c)}\\
& \sum_{(i,j)\in\mathcal{E}^{\text{DARP}}:\ i\in\mathcal{S},j\in\mathcal{S}} x_{i,j} \le |\mathcal{S}| - 2 && \forall \mathcal{S} \in \mathcal{V} : 0 \in \mathcal{S}, 2N+1 \notin \mathcal{S}, && \\
& && \exists i \in \mathcal{P} \cap \mathcal{S} \text{ s.t. } i+N \notin \mathcal{S}, && \text{(1d)}\\
& t_j \ge (t_i + b_i + c_{i,j})\, x_{i,j} && \forall (i,j) \in \mathcal{E}^{\text{DARP}}, && \text{(1e)}\\
& \ell_i \le t_i \le u_i && \forall i \in \mathcal{V}^{\text{DARP}}, && \text{(1f)}\\
& t_{i+N} \le t_i + b_i + c_{i,i+N} + W && \forall i \in \mathcal{P}^{\text{DARP}}, && \text{(1g)}\\
& q_j \ge (q_i + d_i)\, x_{i,j} && \forall (i,j) \in \mathcal{E}^{\text{DARP}}, && \text{(1h)}\\
& \max\{0, d_i\} \le q_i \le \min\{Q, Q + d_i\} && \forall i \in \mathcal{V}^{\text{DARP}}, && \text{(1i)}\\
& x_{i,j} \in \{0,1\} && \forall (i,j) \in \mathcal{E}^{\text{DARP}}. && \text{(1j)}
\end{aligned}
$$

The objective (1a) minimizes the total driving time between nodes. Constraints (1b) and (1c) are flow conservation constraints. Constraints (1d) are precedence constraints that enforce that each pickup should occur before its corresponding dropoff. Constraints (1e) and (1f) are travel time and time window constraints, and constraints (1g) ensures that passengers are diverted at most W time units relative to the time it would take them to make the trip directly. Constraints (1h) and (1i) are capacity constraints. In paratransit applications, it is common for there to be both wheelchair-bound and ambulatory passengers, with capacity constraints on each. In such cases, the formulation is adapted to include a set of capacity constraints for each type of passenger.

Although constraints (1e) and (1h) are nonlinear, they can be linearized using big-M constraints as illustrated by Ropke et al. (2007). Ropke et al. further contrast this big-M formulation with another formulation PDPTW2 that uses cutting planes to reduce the problem size. However, we found that on our problem data, the big-M constraints had better performance.

The solution to the DARP (1) is a set of trips, where requests may be serviced together in the same vehicle if doing so would reduce the total driving time. Of course, the trips must also

respect the time windows and maximum ride time on the individual requests, as well as capacity constraints on the vehicles.

It is challenging to solve the full DARP on realistic instances on hundreds, let alone thousands, of requests, particularly within computation times that are practical for daily planning. As such, we will discuss a clustering framework that partitions the requests in a principled way such that the DARP can easily be solved on each cluster.

A typical view of clustering is to embed points in a feature space, and then to use an algorithm such as k-means to group the points into clusters based on their proximity in this feature space. In our problem setting, points might correspond to requests, and the features might be the pickup and dropoff locations and times. Because of the simplicity of this proximity-based approach, it is often used in practice; for example, Savelsbergh and Sol (1998) cite such an approach as the baseline algorithm used by their package delivery company of study. However, one limitation of such an approach is that it does not capture requests that can profitably be fulfilled together even if their pickup and dropoff locations are not close together; for example, if cost savings come from the closeness of the the trajectories of the requests rather than their endpoints, as in the case of when a short request can be serviced while already well along the way to servicing another significantly longer request.

Our clustering approach will instead use a more general criterion of whether requests can be fulfilled together, building off an approach in Santi et al. (2014). Santi et al. define a pairwise *shareability graph*, where each node i represents a request, and each edge (i, j) represents whether requests i and j can be fulfilled in a single trip while also reducing total travel time.

Computation of the edges can be done by quickly enumerating each of the four trips that can be made with two requests:

$$i \to j \to i+N \to j+N,$$
$$i \to j \to j+N \to i+N,$$
$$j \to i \to i+N \to j+N,$$
$$j \to i \to j+N \to i+N,$$

recalling the earlier definition of a trip as a sequence of pickups and dropoffs such that the vehicle begins and ends empty but is occupied throughout. Each of the four trips can quickly be evaluated for feasibility of capacity constraints and time windows, and if any have total cost lower than $c_{i,i+N} + c_{j,j+N}$, the edge (i, j) is added to the graph. The edge is weighted by the maximum cost saving among the feasible options of the four enumerated trips.

An example of a shareability graph for a set of requests in Figure 1a is shown in Figure 1b. Figure 1a shows fictitious pickup and dropoff locations for three different requests, as well as arcs that correspond to fulfilling each request separately. For simplicity, time windows and capacities are not displayed, but all trips displayed in Figure 1 are assumed to be feasible. The corresponding shareability graph is shown in Figure 1b, where the edge between nodes 1 and 2, for example, indicate that these two requests could be fulfilled together in a shared trip and save 0.84 units of travel time. The optimal trip for requests 1 and 2 is shown in Figure 1c.

As discussed by Santi et al., by computing the *maximum-weight matching* on the shareability graph, one can obtain the set of trips with lowest cost (maximum cost savings) when considering *only* sharing between two requests. Of course, restricting attention to sharing between pairs of requests is suboptimal when considering higher-capacity vehicles such as vans, which could potentially fulfill several requests at once. In the example of Figure 1, the optimal solution would fulfill all three requests together in a single trip, which is not captured in the matching on the shareability graph. Although it is possible to construct a hypergraph that evaluates general tuples of requests, the computational expense rises dramatically; even for three requests, sixty potential trips must be evaluated.

Nonetheless, from Santi et al. (2014), we see that limited enumeration and graph matching can quickly generate an initial solution of high quality. Meanwhile, column-generation-based approaches such as Savelsbergh and Sol (1998) illustrate the power of iterative methods to improve solution quality over time. We propose a single procedure that combines these useful properties to efficiently achieve cost savings from higher-capacity vehicle sharing. The first step is identical to that in Santi et al. (2014), where we compute a maximum-weight matching between requests on the shareability graph and output the associated trips.

Subsequent steps of our procedure continue as follows. We can construct a condensed version of the shareability graph, where each node now represents a trip computed in the previous step. In the updated graph, an edge exists between trip nodes i and j if there exists any edge in the original shareability graph between a request from trip i and a request from trip j. The edge weight between trip i and trip j is taken as the maximum of the edge weights from the original shareability graph over all edges between requests from trip i and requests from trip j. For example, the shareability graph computed on the trips from Figure 1c is shown in Figure 1d.

We can then compute the maximum-weight matching on the new shareability graph, and solve an instance of the DARP on each matched pair of trip nodes. Each instance of the DARP considers only the requests for the two trips in question, and is warm-started using the trips computed from the previous iteration. This step is illustrated in Figures 1d and 1e, recalling that the first step produced the trips in Figure 1c. In the case of Figure 1d, requests 1 and 2 are combined into a

supernode representing their shared trip; an edge between the supernode and request 3 corresponds to the edge between requests 1 and 3 in the original shareability graph of Figure 1b. The DARP would be solved on all three requests, warm-started with the trips of Figure 1c, and would produce the trip in Figure 1e before terminating.

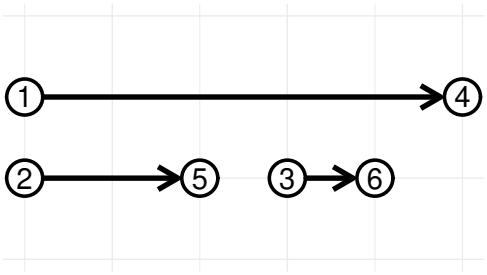
Alonso-Mora et al. (2017) also extended the work of Santi et al. (2014) to higher-capacity vehicle sharing by solving routing problems on cliques in a modified shareability graph; the drawback that we found with this approach was that on our datasets, the size and density of the modified shareability graphs led to an impractically high number of cliques. For example, on a smaller 3,494-request weekend example, the shareability graph had 60,639 edges and produced 122,330 cliques, each leading to a different small instance of the DARP, and could not be solved within our desired timeframe. By contrast, our algorithm involved 2,975 small instances of the DARP.

Our overall algorithm can be thought of one that makes local improvements that are guided by insight from optimization (i.e., the maximum-weight matching). High-quality solutions can be found efficiently by solving DARPs on gradually increasing problem sizes, aided by the strong warm starts provided by the trips in the previous iteration. Of course, part of the computational efficiency comes from the fact that the edge weights in the shareability graph are computed once in the beginning and are not updated thereafter, leading to inaccuracy in subsequent iterations. As a result, some instances of the DARP may not lead to improved solutions. The tradeoff between extra computation of improved edge weights versus saved computation from fewer instances of the DARP is one that could be further explored. However, as we will show in our computational results, the existing framework shows strong performance, producing near-optimal solutions on medium graphs and high-quality solutions on large graphs in seconds to minutes.

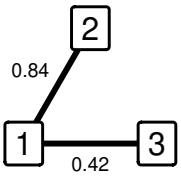
3.3. Connecting Trips

The output of the clustering procedure outlined in Section 3.2 is a set of trips (as defined at the beginning of Section 3). The second and final phase of the paratransit routing process is to take this set of trips, and connect them into an itinerary for each driver that he can fulfill during his shift. We refer to this problem as the *trip connection problem* (TCP).

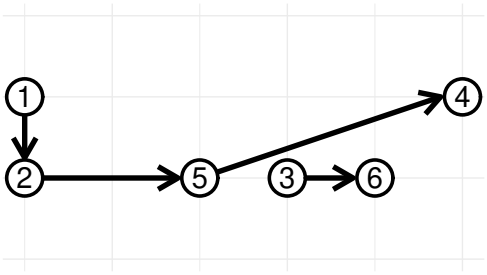
3.3.1. Base Formulation The base formulation for the problem of connecting a set of A trips is a special case of the DARP in which each trip corresponds to a "request" and trips do not share the vehicle at the same time but must be fulfilled individually. As such, we use analogous notation as the input data to the DARP (1) to make clear the similarities. In this special case, the pickup nodes are the starting node (first pickup) of each trip, dropoff nodes are the ending node (last dropoff) of each trip, each pickup node only has one outgoing edge to its corresponding



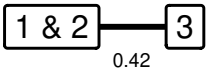
(a) Pickup (1, 2, 3) and dropoff (4, 5, 6) locations for three fictitious requests. The arcs show each request being fulfilled in three separate trips.



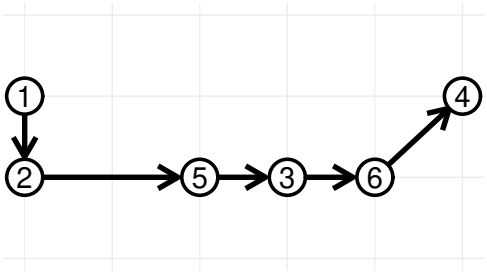
(b) The shareability graph constructed from the requests in Figure 1a.



(c) Two trips generated after a single iteration of the algorithm.



(d) The shareability graph constructed from the trips in Figure 1c.



(e) All three requests are combined into a single trip after two iterations of the algorithm.

Figure 1 An illustration of the algorithm for clustering requests into trips on three fictitious requests.

dropoff node, and dropoff nodes are only connected to pickup nodes for other requests. We use the notation $\mathcal{P}^{\text{TCP}}$ to refer to the set of trip starting nodes, $\mathcal{D}^{\text{TCP}}$ to refer to the set of trip ending nodes, and $\mathcal{V}^{\text{TCP}}$ to refer to the set of all nodes, including the source (0) and sink $(2A+1)$. Then, the edge set $\mathcal{E}^{\text{TCP}}$ is defined as follows:

$$\begin{aligned}\mathcal{E}^{\text{TCP}} = &\{(0,i)\}_{i\in\mathcal{P}^{\text{TCP}}} \cup \{(i,2A+1)\}_{i\in\mathcal{D}^{\text{TCP}}} \cup \\ &\{(i,i+A)\}_{i\in\mathcal{P}^{\text{TCP}}} \cup \{(i,j)\}_{i\in\mathcal{D}^{\text{TCP}},j\in\mathcal{P}^{\text{TCP}},j\neq i-A}. \end{aligned} \tag{2}$$

The service duration and time window input data must also be modified slightly. Because a feasible trip must satisfy the time windows for all of the requests it contains simultaneously, a time

window $[\ell_i, u_i]$ for $i \in \mathcal{V}^{\mathrm{TCP}}$ may actually be smaller than the time window of the original request corresponding to node i. Accordingly, the formulation of the TCP should use these smaller time windows. The service duration b_i at pickup node $i \in \mathcal{P}^{\mathrm{TCP}}$ should be defined as the shortest time to fulfill the trip from start to end, plus the load time at the start and including intermediate load and unloading times, minus $c_{i,i+A}$. At dropoff node $i \in \mathcal{D}^{\mathrm{TCP}}$, the service duration is the unloading time, as before. With these modifications, constraint (1f) remains correct. A complication arises if the time to fulfill the trip is different depending on where in the time window the pickup and dropoff times fall. To avoid infeasibility, one could either take the largest of these values, or add cutting planes to eliminate any sequences of trips that do not satisfy all time windows.

Having made the appropriate modifications to the input data definitions, we reproduce the problem below as it applies to routing A trips rather than N requests. Recall that trips are satisfied one at a time, and each trip already respects capacity constraints and precedence constraints; therefore, constraints (1d), (1g), (1h), and (1i) are omitted from the new formulation. The decision variable $z_{i,j}$ for $(i,j) \in \mathcal{E}^{\mathrm{TCP}}$ represents whether some vehicle travels from node i to node j (an analogue of $x_{i,j}$ in the DARP (1)), and the decision variable y_i for $i \in \mathcal{V}^{\mathrm{TCP}}$ represents the arrival time at node i (an analogue of $t_{i,j}$ in the DARP (1)). Then, trips are connected into itineraries of minimum cost by solving the following optimization problem:

$$
\begin{aligned}
(\mathrm{TCP}) \quad \min_{\mathbf{z},\mathbf{y}} \quad & \sum_{(i,j)\in\mathcal{E}^{\mathrm{TCP}}} c_{i,j} z_{i,j} && && \text{(3a)}\\
\text{s.t.} \quad & z_{i,i+A} = 1 && \forall i \in \mathcal{P}^{\mathrm{TCP}}, && \text{(3b)}\\
& \sum_{j\in\mathrm{Out}(i)} z_{i,j} = 1 && \forall i \in \mathcal{D}^{\mathrm{TCP}}, && \text{(3c)}\\
& \sum_{i\in\mathrm{In}(j)} z_{i,j} = 1 && \forall j \in \mathcal{V}^{\mathrm{TCP}}, && \text{(3d)}\\
& y_j \geq (y_i + b_i + c_{i,j})\, z_{i,j} && \forall (i,j) \in \mathcal{E}^{\mathrm{TCP}}, && \text{(3e)}\\
& \ell_i \leq y_i \leq u_i && \forall i \in \mathcal{V}^{\mathrm{TCP}}, && \text{(3f)}\\
& z_{i,j} \in \{0,1\} && \forall (i,j) \in \mathcal{E}^{\mathrm{TCP}}. && \text{(3g)}
\end{aligned}
$$

The objective (3a) minimizes the driving time (again excluding any wait time between trips). Constraints (3b), (3c), and (3d) are flow conservation constraints, recalling that the trip pickup nodes must be immediately followed by the trip dropoff nodes. Constraints (3e) and (3f) are travel time and time window constraints. As before, although constraint (3e) is nonlinear, it can be linearized using big-M constraints, as before. A smaller formulation that collapses $\mathcal{P}^{\mathrm{TCP}}$ and $\mathcal{D}^{\mathrm{TCP}}$ into a single node is also possible (Bertsimas et al. 2019b), but we use the larger formulation to illustrate the connection with the DARP (1). The output of TCP (3) will be a set of driver itineraries (sequences of trips).

3.3.2. Adding Labor Constraints As stated in Problem (3), the TCP does not as yet account for labor constraints, which are a critical component of paratransit operations. Paratransit agencies may operate multiple fleets of heterogeneous vehicles of varying capacities, with each fleet operated by a different service provider. Service providers can be contracted to fulfill overlapping subsets of requests; for example, they may share the central urban core and partition the outlying suburbs geographically. Driver shifts can also specified for each day of the week, with each shift associated with a start time, an end time, a vehicle type, and a service provider.

The addition of labor constraints to the TCP (3) necessitates two major changes to the formulation. First, if drivers are scarce, the problem may not be feasible. To ensure feasibility, we add an option to outsource requests to TCPs, which is the policy employed in Boston. We use the decision variable w_i for $i = 1, \ldots, A$ to indicate whether the requests of trip i are outsourced, and add a cost coefficient λ_i for $i = 1 \ldots, A$ to represent the cost of outsourcing a trip. The cost λ_i is flexible, and can be chosen to be proportional to the number of requests or the total travel distance in trip i, for example. In cases where outsourcing is not a policy of the paratransit agency, one should set λ_i to be an appropriately large value so as to discourage outsourcing. If there are any trips that cannot fit in the itineraries, the requests in these trips could be inserted as a post-processing step.

The second change to the formulation addresses the heterogeneity due to the different vehicle types (e.g. van, sedan), service providers, and driver shift start and end times. If the number of different vehicle type-provider-shift tuples is small, the heterogeneity can be modeled by expanding the $\mathbf{y}$ and $\mathbf{z}$ variables from two to three indices, with the third index indicating the vehicle type-provider-shift. However, in practice, the number of tuples can number in the hundreds or more, which rendering a three-index model intractable. We will strike a balance between expansion and aggregation by using the third index to indicate the vehicle type-provider, and treating the driver shift start and end times in an aggregated way.

The vehicle type-providers will be indexed by $k = 1, \ldots, K$. The notation $\mathcal{A}_k \subset \{1, \ldots, A\}$ will be used to indicate the subset of trips that can be satisfied by a vehicle of type-provider k, and $\mathcal{A}_k(t) \subset \mathcal{A}_k$ will be used to indicate the subset of trips that must be fulfilled during time t. Namely, $\mathcal{A}_k(t)$ is defined as follows:

$$\mathcal{A}_k(t) = \{i \in \mathcal{A}_k \mid u_i \leq t, \ell_{i+A} \geq t\}.$$

$\mathcal{A}_k(t)$ can be defined for all t in the set of unique shift start and end times, which we will denote as $\mathcal{T}$. Similarly, V_k will denote the total number of vehicles of type-provider $k = 1, \ldots, K$ available at time t, and $V_k(t)$ for $t \in \mathcal{T}$ will denote the total number of vehicles of type-provider k available at time t.

The node set $\mathcal{V}_k^{\text{TCP}} \subset \mathcal{V}^{\text{TCP}}$ for $k = 1, \ldots, K$ is defined as

$$\mathcal{V}_k^{\text{TCP}} = \{0, 2A+1\} \cup \mathcal{A}_k \cup \{i + A \mid i \in \mathcal{A}_k\}, \tag{4}$$

and the edge set $\mathcal{E}_k^{\text{TCP}}$ for $k = 1, \ldots, K$ is defined as

$$\mathcal{E}_k^{\text{TCP}} = \{(i,j) \in \mathcal{E}^{\text{TCP}} \mid i \in \mathcal{V}_k^{\text{TCP}}, j \in \mathcal{V}_k^{\text{TCP}}\}. \tag{5}$$

The decision variable $z_{i,j}^k$ for $(i,j) \in \mathcal{E}_k^{\text{TCP}}$ and $k = 1, \ldots, K$ now indicates whether a vehicle of type-provider k travels from node i to node j, and the decision variable y_i^k for $i \in \mathcal{V}_k^{\text{TCP}}$ and $k = 1, \ldots, K$ now indicates the arrival time of a vehicle of type-provider k at node i.

With these modifications, the formulation for the TCP with labor constraints, which we call TCP-Labor, is as follows:

$$\text{(TCP-Labor)} \quad \min_{\mathbf{z},\mathbf{y},\mathbf{w}} \quad \sum_{(i,j)\in\mathcal{E}^{\text{TCP}}} \sum_{k=1}^{K} c_{i,j} z_{i,j}^k + \sum_{i=1}^{A} \lambda_i w_i \tag{6a}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in\mathcal{E}_k^{\text{TCP}}} z_{i,j}^k = \sum_{j:(j,i)\in\mathcal{E}_k^{\text{TCP}}} z_{j,i}^k \qquad \forall i \in \mathcal{V}_k^{\text{TCP}},\ k = 1, \ldots, K, \tag{6b}$$

$$\sum_{k=1}^{K} \sum_{j:(i,j)\in\mathcal{E}_k^{\text{TCP}}} z_{i,j}^k = 1 - w_i \qquad \forall i \in \mathcal{P}^{\text{TCP}}, \tag{6c}$$

$$y_j^k \geq \left(y_i^k + b_i + c_{i,j}\right) z_{i,j}^k \qquad \forall (i,j) \in \mathcal{E}_k^{\text{TCP}},\ k = 1, \ldots, K, \tag{6d}$$

$$\ell_i \leq y_i^k \leq u_i \qquad \forall i \in \mathcal{V}_k^{\text{TCP}},\ k = 1, \ldots, K, \tag{6e}$$

$$\sum_{i\in\mathcal{A}_k} z_{0,i}^k \leq V_k \qquad \forall k = 1, \ldots, K, \tag{6f}$$

$$\sum_{i\in\mathcal{A}_k(t)} \sum_{j:(i,j)\in\mathcal{E}_k^{\text{TCP}}} z_{i,j}^k \leq V_k(t) \qquad \forall k = 1, \ldots, K,\ t \in \mathcal{T}, \tag{6g}$$

$$z_{i,j}^k \in \{0,1\} \qquad \forall (i,j) \in \mathcal{E}_k^{\text{TCP}},\ k = 1, \ldots, K, \tag{6h}$$

$$w_i \in \{0,1\} \qquad \forall i = 1, \ldots, A. \tag{6i}$$

Constraint (6b) is a combination of constraints (3b), (3c), and (3d), but does not necessarily require that all nodes see nonzero inflow and outflow. Constraint (6c) indicates whether trip i is outsourced using the indicator variable w_i, which is set to 1 if the trip sees zero flow. Constraints (6d) and (6e) are time window constraints analogous to constraints (3e) and (3f). Constraints (6f) and (6g) limit the vehicles used to only those that are available in total and at each time point. Further constraints could be imposed on $\mathbf{w}$ to address concerns such as requests that cannot be outsourced due to additional needs of the passengers, or limiting the amount of total outsourcing.

As mentioned before, the solutions to TCP-Labor may not necessarily be feasible for the specified shifts because TCP-Labor (6) deals with the shift times in aggregation. For example, a solution

itinerary may actually stretch across two separate shifts (say, a shift from 8am to 3pm and another from 3pm to 10pm). Postprocessing of the solutions is required to produce feasible driver itineraries, and we propose a simple and fast approach. First, we take each itinerary produced by the model and break it into all possible sub-itineraries, down to the individual trips that were connected together. Then, we can use these sub-itineraries as input to a set cover problem that is traditionally used to in column generation approaches (Savelsbergh and Sol 1998, Bertsimas et al. 2019a).

TCP-Labor can also be seen as a way of quickly generating a large number of high-quality columns. Agencies with more computational resources or time available could consider solving TCP-Labor with a range of λ_i parameters, thus generating even more variation among the columns produced.

4. Paratransit Decision-Making in Practice

4.1. Background of Operations

The RIDE is a paratransit agency that is operated by the Massachusetts Bay Transportation Authority (MBTA) and provides door-to-door shared rides to the greater Boston area. It typically sees about 7,000 requests on weekdays and 3,000 requests on weekends, which are the largest problem sizes we have seen in the literature. Requests for transportation must be submitted one to seven days in advance, which gives enough time for driver routes to be computed and pickup times to be communicated to passengers the night before.

Each request is categorized internally as either a pickup or an appointment, with a 20-minute time window allowed for pickups and a 40-minute time window allowed for appointments. Passenger travel times are mandated to be at most 30 minutes more than the travel time experienced on public transit. In our algorithm, we imposed the stricter requirement that passengers should experience travel times at most 30 minutes more than that of a direct route by taxi (as opposed to by public transit), which should provide passengers a higher-quality service.

In addition to time windows and maximum ride times, requests are also associated with passenger types (wheelchair seats and/or ambulatory seats) and service animals. The characteristics of the request determine what kind of vehicle is needed to service the request. Requests that involve wheelchair-bound passengers, service animals, or more than three ambulatory passengers must be serviced by vans; all other requests can be serviced by sedans unless they are serviced along with enough other requests that a van is needed for its capacity. In the labor data provided to us, between 298 and 616 vehicles were available each day, with shift lengths ranging from four to twelve hours.

The vehicles are operated by three different vendors, which partition the greater Boston area into service regions and share the central urban core. The individual service regions are called North,

West, and South, for their geographic locations relative to Boston. For example, any requests within Newton, in the West region, are serviced by a different provider than those within Quincy, in the South region. For requests between individual service regions, such as those from Newton to Quincy or vice versa, the passengers are dropped off at a pre-specified transfer point between vendors. The intention of this partitioning is to ensure that drivers are not taken too far away from their home garage, so as to limit the amount of time spent driving empty. The partitions and transfer points are set in advance and are exogenous to our model, representing a constraint rather than a decision to be optimized.

The RIDE has recently begun outsourcing some requests to TNCs such as Uber and Lyft. Despite the high costs of paratransit, TNCs are unlikely to fully replace paratransit due to safety and privacy concerns; however, limited and judicious outsourcing can certainly help defray paratransit operating costs.

Given that requests for transportation must be submitted at least one day in advance, it may appear that an algorithm that there are several hours available over the course of the previous night for computation time. However, in practice, computation time is much scarcer; practitioners may need additional time to modify or execute a plan (or sleep). As such, our goal was to develop an algorithm that could run in half an hour or less on Boston's data, allowing practitioners plenty of time in case the algorithm needed to be run again.

4.2. Day-Of Disruptions

Our focus thus far has been on addressing the deterministic optimization problem; however, The RIDE's operations in actuality do experience significant uncertainty. A major source of uncertainty comes from cancellations, where passengers may opt to cancel their rides after The RIDE has contacted them with their scheduled pickup and dropoff times; on average, about 20% of requests are cancelled every day. Although a reasonable strategy might be simply to skip the cancelled requests, such a strategy might result in significantly less efficient routes, particularly if requests of longer travel distance are cancelled. This was another motivation for developing an algorithm with modest running times, so that it could be adapted for re-optimization.

4.3. Evaluating Paratransit Performance

The RIDE uses many metrics to measure its performance. Chief among these metrics is *productivity*, which is defined as the number of requests served per hour. The denominator is measured as the total time that vehicles spend on the road between first pickup and last dropoff. At the time of this writing, drivers for The RIDE are paid only for the hours between first pickup and last dropoff, which are called *revenue hours*. As such, productivity is a key driver of expenses at The RIDE.

However, the metric of *total hours* spent from garage to garage is naturally important to the driver, since this represents the entire workday experienced by the driver. Another metric, *utilization*, is defined as the ratio of revenue hours to total hours, and is intended to capture the driver's desire to spend the bulk of his or her time generating revenue.

4.4. A Baseline Insertion Heuristic

We do not know the details of the algorithm that The RIDE currently uses for its operations. As a baseline, we implemented an algorithm described in Marković et al. (2015) that is based on insertion heuristics. At a high level, this algorithm considers each request in order of increasing pickup time, and for each request evaluates the potential vehicles that could service the request before choosing the vehicle of lowest insertion cost. Marković et al. (2015) also describe some local exchanges that may be performed to further reduce costs, which we did not implement because the base algorithm proved to be quite computationally intensive on our problem sizes of several thousand requests.

The algorithm from Marković et al. (2015) has been implemented in Maryland, and when we implemented it ourselves, we found that its performance metrics were comparable to those reported by The RIDE. These two points lead us to suspect that the algorithm at The RIDE is likely based on insertion heuristics as well.

5. Computational Results

In this section, we evaluate the performance of our algorithm on real-world data provided by The RIDE, the paratransit agency operated by the Massachusetts Bay Transportation Authority (MBTA) in the greater Boston area. We were given thirty days' worth of data between August and September 2018. For evaluation, we ran our algorithm on a week of data from September 10, 2018 to September 16, 2018. The week was chosen arbitrarily, but happens to correspond to a particularly high-volume week.

In our experiments, advised by The RIDE, we set driving speeds to 8.3 miles per hour in the core urban area and 13.5 miles per hour in outlying areas over weekdays, and 9.8 miles per hour in the core urban area and 16.3 miles per hour in outlying areas over weekends. These values are conservative, particularly during off-peak hours when roads are less congested. When we constructed trips by solving the DARP (1), we set the wheelchair and ambulatory capacities to van levels. Often, the van capacity constraints were not binding; in our solutions, 36 to 41% of trips in the solutions required vans, which comprised 64 to 69% of the vehicles available.

Our algorithm was implemented using the `Julia` language (Bezanson et al. 2014) and the optimization package `JuMP` (Lubin and Dunning 2015) using the Gurobi solver v8.1 (Gurobi Optimization, Inc. 2016).

The remainder of the computational section is outlined as follows. In Section 5.1, we evaluate the performance of our clustering approach.

5.1. Evaluating Clustering

In the following experiment, we evaluate the performance of the clustering methodology proposed in Section 3.2 on medium- to large-sized graphs. Random requests in subsets of varying sizes from 50 to 1,600 were selected from the weekday data, between 9am and 3pm in the urban core. A typical weekday would have about 1,200 requests in this time frame and area. Ten sets of random requests were generated for each problem size. The times and area were chosen since they represent peak demand, when vehicle-routing problems should be harder to solve.

Shareability graphs were constructed on the random requests, and requests were clustered and routed into trips according to the procedure outlined in Section 3.2, capped at fifteen iterations. The objective value from our procedure was then compared to that obtained to a local search algorithm implemented with Google OR-Tools (2019), as well as to that of the single-iteration matching procedure of Santi et al. (2014). On the larger problem sizes, Google OR-Tools could run for an extremely long time; we capped its running time to be roughly three times that taken for our procedure in order to see what solutions might be produced in somewhat comparable running times. We also report optimality gaps for problem sizes up to $N = 200$, with the optimal solutions for each problem instance computed using enumeration. For the $N = 200$ instances, the edge sets $\mathcal{E}^{\text{DARP}}$ were sparsified to include only the 50 nearest neighbors of each node; still, enumeration could take two hours or more for each problem size due to the scale and the flexibility of the problem parameters such as time windows. We chose to compute the gaps through enumeration rather than solving a MIO problem since the gaps from Gurobi could be quite large even on small problem sizes, possibly because of the lazy constraints (1d). The enumeration instances were scheduled as jobs on a cluster with Intel Xeon E5-2650 cores; each was given 4GB of memory. The remainder of the experiments in this section were run on a laptop with an Intel i7-6500U processor and 16GB of RAM.

Results from the various approaches on the random problem instances of varying sizes are summarized in Table 2 and Figure 2. The first column of Table 2 shows the number of requests sampled, the second column shows the method used, the third column shows the mean objective (1a) value in minutes, the fourth column shows the optimality gap for the problem instances for which the optimal solution could be computed, the fifth column shows the number of trips that were in the optimal solution, and the final column shows the running time in seconds (including the time to build the shareability graph). Figure 2 shows the performance of the various approaches against problem size. On the y-axis, the objective value of the best solution ("Shared Obj.") is divided by

the objective value of the solution where all requests are fulfilled individually ("Unshared Obj."), so that all problem sizes may be plotted on the same graph. Note that because the objective is to minimize cost, lower on the graph is better, and that the y-axis values should decrease as number of requests increase since more requests should give more opportunities for improvement from sharing vehicles.

From Table 2 and Figure 2, we can make a few observations around performance in objective value. Firstly, on all problem sizes, our method shows the strongest performance, with improvement of 10.4% over Santi et al. (2014) and 19.6% over Google OR-Tools on the largest graphs. Secondly, on the smaller problem sizes from $N = 50$ to $N = 200$, our method produces near-optimal solutions with optimality gaps of small fractions of a percent. Thirdly, although Google OR-Tools shows strong performance at the lower problem sizes, it is unable to find high-quality solutions in reasonable running times at $N = 800$ and $N =$1,600, illustrating that even though metaheuristics are perceived as scalable algorithms, they too can be overwhelmed by large problem sizes. By contrast, both our method and Santi et al. (2014) show stable performance at the largest problem sizes, illustrating the power of a limited amount of enumeration to guide optimization towards high-quality solutions. Of course, metaheuristics can benefit from more careful parameter tuning, and it is possible that such an approach could further improve Google OR-Tools. However, this offers yet another point of contrast to our methodology; ours requires no tuning, and thus can be expected to have stable performance on different datasets.

An additional benefit to our method is that on the largest problem sizes, the number of trips produced (A) is lowered substantially, which will lead to smaller problem sizes in the second step of connecting trips into itineraries. Finally, although our approach naturally has a longer running time than that of Santi et al. (2014), the running times continue to be modest, only about four minutes for the largest problem instances of $N =$ 1,600 requests, which could easily be sped up with parallelization.

We next use one week of real data to compare the performance of the various approaches, with the exception of Google OR-Tools (2019), for which we were unable to obtain high-quality results in reasonable running times. The results are summarized in Table 3. Overall, our method improves the objective by about 3% on weekends and 7% on weekdays, and reduces the number of trips by about 10% on weekends and about 16% on weekdays. The improvement is lower than in the random graphs of Table 2; this is because the most substantial gains from vehicle sharing are to be found when demand is dense, while the entire dataset includes areas and times of sparser demand. Finally, both methods have acceptable running times for daily planning. Although our method naturally takes longer, the running times of about two minutes for weekends and several minutes for weekdays are quite modest, and could be sped up even further by parallelization.

To summarize, our computational results on our clustering procedure have shown the following:

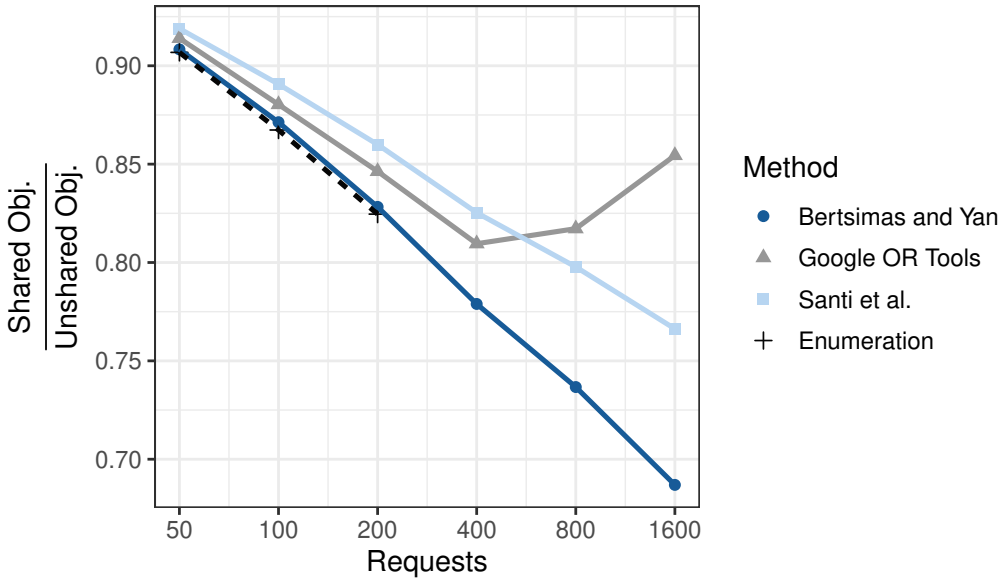
Table 2 Performance summary for three different methods of clustering and routing N requests into A trips on random problem instances sampled from real data. For each problem size N, the performance metrics are reported over ten simulated graphs. For larger problem sizes $N \geq 400$, we were unable to compute the optimal solution, so we omit the optimality gap.

Requests N	Method	Obj. (min)	Opt. Gap	Trips A	Run. Time (s)
50	Bertsimas and Yan	**1,143**	**0.2%**	39	0
	Santi et al.	1,157	1.3%	41	**0**
	Google OR Tools	1,150	0.8%	**35**	1
100	Bertsimas and Yan	**2,169**	**0.5%**	72	1
	Santi et al.	2,217	2.7%	76	**0**
	Google OR Tools	2,191	1.5%	**63**	3
200	Bertsimas and Yan	**4,289**	**0.4%**	128	2
	Santi et al.	4,453	4.3%	143	**0**
	Google OR Tools	4,382	2.6%	**109**	9
400	Bertsimas and Yan	**7,835**	–	**226**	8
	Santi et al.	8,302	–	269	**1**
	Google OR Tools	8,143	–	228	26
800	Bertsimas and Yan	**14,835**	–	**407**	26
	Santi et al.	16,064	–	508	**5**
	Google OR Tools	16,456	–	596	81
1,600	Bertsimas and Yan	**27,901**	–	**725**	85
	Santi et al.	31,123	–	971	**20**
	Google OR Tools	34,704	–	1,369	258

Table 3 Performance summary for various methods of clustering and routing N requests into A trips on one week's worth of real data.

Date	Requests N	Method	Obj. (hr)	Trips A	Run. Time (min)
Mon, Sep 10	6,600	Bertsimas and Yan	**2,363**	**3,396**	9.6
		Santi et al.	2,528	4,136	**3.5**
Tue, Sep 11	6,783	Bertsimas and Yan	**2,428**	**3,520**	10.1
		Santi et al.	2,598	4,244	**3.4**
Wed, Sep 12	7,437	Bertsimas and Yan	**2,608**	**3,786**	11.7
		Santi et al.	2,796	4,625	**4.3**
Thu, Sep 13	7,408	Bertsimas and Yan	**2,610**	**3,792**	11.5
		Santi et al.	2,795	4,608	**4.2**
Fri, Sep 14	7,026	Bertsimas and Yan	**2,466**	**3,655**	10.4
		Santi et al.	2,633	4,402	**3.7**
Sat, Sep 15	3,494	Bertsimas and Yan	**1,174**	**2,216**	2.5
		Santi et al.	1,212	2,457	**0.6**
Sun, Sep 16	3,096	Bertsimas and Yan	**1,049**	**1,902**	2.2
		Santi et al.	1,091	2,154	**0.6**

Figure 2 (Color online) Performance of method in Section 3.2 against Google OR-Tools (2019), Santi et al. (2014), and enumeration on problem instances of varying size.



- Near-optimal performance on medium-sized graphs sampled from real data, and
- Scalability to large problem sizes comprising thousands of requests, which cause trouble even for a competitive local search procedure.

These two characteristics are achieved by leveraging a framework that solves multiple tiny instances of the DARP (1) before iteratively building up the problem sizes, using the solutions obtained in previous iterations as strong integer-optimization warm starts.

5.2. Evaluating Trip Connection

Having evaluated the clustering method from Section 3.2, we now turn to evaluating the step of connecting trips into driver itineraries, as described in Section 3.3. In this next set of experiments, we evaluate the performance and running time of our procedure on a week of data, and report key metrics used by paratransit agencies from Section 4.3.

In the problem instances that follow, we set an outsourcing cost of $w_i = 100$, with all costs measured in minutes. This high outsourcing cost was chosen to create a more challenging and therefore interesting test case.

The trips summarized in Table 3 are still numerous and lead to large problem sizes of over 100,000 variables on even the smaller weekend cases. To reduce the problem size, we connected trips to nearest neighbor trips if they were less than ten minutes apart in driving distance, with at most five minutes of empty waiting time between trips. On the trips generated from our method in Section 3.2, this reduced the trip sets to about 2,300 trips on weekdays and 1,300 trips on weekends.

The problem sizes for the reduced trip sets are shown in Table 4, both for the trips generated from our method as well as for the trips generated from Santi et al. (2014). On average, our clustering method produced TCP (6) instances about 18% smaller than Santi et al. (2014).

Although such an approach might naturally introduce some suboptimality, these reduced problem sizes allow us to find solutions within reasonable running times. On the full trip set for the smaller weekend cases, we were unable to find high-quality solutions within the five minutes that we allotted to solve the TCP (6), and even after an hour of computation time we were unable to improve upon the performance of the reduced solutions.

Table 4 Model sizes for TCP (6) on different clustered trip instances from one week's worth of real data. "Cluster BY-BY" refers to the TCP (6) run on the trips generated according to Section 3.2. "Cluster S-BY" refers to the TCP (6) run on the trips generated using Santi et al. (2014).

Date	**Requests** N	**Method**	**Trips** A	**Variables**	**Constraints**
Mon, Sep 10	6,600	Cluster BY-BY	**2,169**	**79,815**	**16,965**
		Cluster S-BY	2,396	101,091	20,642
Tue, Sep 11	6,783	Cluster BY-BY	**2,244**	**82,091**	**17,056**
		Cluster S-BY	2,440	101,851	21,642
Wed, Sep 12	7,437	Cluster BY-BY	**2,425**	**89,894**	**18,762**
		Cluster S-BY	2,650	112,242	22,839
Thu, Sep 13	7,408	Cluster BY-BY	**2,376**	**87,921**	**17,816**
		Cluster S-BY	2,617	112,755	21,221
Fri, Sep 14	7,026	Cluster BY-BY	**2,326**	**91,584**	**19,045**
		Cluster S-BY	2,554	110,566	23,175
Sat, Sep 15	3,494	Cluster BY-BY	**1,402**	**37,548**	**9,836**
		Cluster S-BY	1,464	41,672	10,163
Sun, Sep 16	3,096	Cluster BY-BY	**1,238**	**32,541**	**8,420**
		Cluster S-BY	1,310	39,363	9,826

Performance metrics for our algorithm are summarized in Table 5. For comparison, our full procedure ("Cluster BY-BY") is compared against the TCP (6) run on the trips generated using Santi et al. (2014), as well as the insertion heuristic described in 4.4. Relative to the insertion heuristic, which is our approximation of the algorithm currently used at The RIDE, we improve productivity by 30 to 40%. The gains of about 3% on weekends and 7% on weekdays that we saw in the clustering step relative to Santi et al. (2014) lead to improvements in productivity by about 6% on weekends and 11% on weekdays. This suggests that any improvements in the first clustering step could be compounded in the connection step, possibly because when there are fewer trips, fewer empty vehicle hours are needed to connect them.

All three methods keep outsourcing roughly at the desired threshold of 10% of requests or less, with our clusters outsourcing 7.4%, the Santi et al. (2014) clusters outsourcing 7.6%, and the insertion heuristic outsourcing 10.8%. Productivity generally should increase as the number of requests outsourced increases, since ill-fitting requests will not be forced into driver itineraries. Therefore, it is notable that our full procedure achieves higher productivity metrics while outsourcing fewer requests than the other two approaches.

Table 5 Performance summary for TCP (6) on different clustered trip instances from one week's worth of real data, as well as an insertion heuristic. "Cluster BY-BY" refers to the TCP (6) run on the trips generated according to Section 3.2. "Cluster S-BY" refers to the TCP (6) run on the trips generated using Santi et al. (2014). The insertion heuristic is similar to that implemented by Marković et al. (2015).

Date	**Reqs.** N	**Method**	**Productivity** (reqs./hr)	**Outsourced** (reqs.)	**Utilization**
Mon, Sep 10	6,600	Cluster BY-BY	**2.03**	503	0.83
		Cluster S-BY	1.84	**481**	0.85
		Insertion heuristic	1.46	739	**0.88**
Tue, Sep 11	6,783	Cluster BY-BY	**2.03**	**440**	0.83
		Cluster S-BY	1.83	490	0.85
		Insertion heuristic	1.47	733	**0.88**
Wed, Sep 12	7,437	Cluster BY-BY	**2.09**	594	0.83
		Cluster S-BY	1.88	**566**	0.86
		Insertion heuristic	1.54	866	**0.88**
Thu, Sep 13	7,408	Cluster BY-BY	**2.07**	**532**	0.84
		Cluster S-BY	1.87	544	0.86
		Insertion heuristic	1.54	828	**0.88**
Fri, Sep 14	7,026	Cluster BY-BY	**2.07**	**553**	0.84
		Cluster S-BY	1.86	611	0.85
		Insertion heuristic	1.53	876	**0.88**
Sat, Sep 15	3,494	Cluster BY-BY	**1.98**	**287**	0.87
		Cluster S-BY	1.88	293	0.87
		Insertion heuristic	1.52	282	**0.90**
Sun, Sep 16	3,096	Cluster BY-BY	**2.02**	198	0.85
		Cluster S-BY	1.89	197	0.86
		Insertion heuristic	1.49	**196**	**0.89**

Finally, the running times for our methods are shown in Table 6. For the cluster-based methods, the total running time is broken into the first clustering step and the second trip connection step. The total also includes some extra processing time that is not included in the "Cluster" and "Connect" columns, which is why the "Total" column is greater than the sum of the previous columns. For the trip connection step, a time limit of five minutes was imposed on Gurobi. Most of the weekday instances on the Santi et al. (2014) clusters did not solve to optimality, but the

maximum optimality gap we obtained was a modest 0.03%. In all cases where the running time was less than five minutes, the model was solved to optimality.

For the insertion heuristic, running times are reasonable for daily use, with of several minutes for weekends and about half an hour for weekdays. However, the heuristic still lags our method in total running time because of the computational expense incurred from evaluating a large number of potential insertions per request. As discussed before in the results of Table 3, our clustering method naturally takes longer than that of Santi et al. (2014) because of the increased iteration count. However, our clustering method also produces smaller model sizes, which then leads to reduced running times in the trip connection step. The total increase in running time for our clusters versus those from Santi et al. (2014) is then modest, amounting to only two or three minutes in running time. Regardless of which clusters are used, the reported running times of less than twenty minutes for weekdays and less than five minutes for weekends are practical for daily use, and could be sped up even further by parallelizing the first clustering step.

To summarize, we have presented a tractable cluster-then-route algorithm based on integer optimization that substantially improve on the algorithms currently used in practice. When run on clusters from Santi et al. (2014), our method improves upon the insertion heuristic by 21 to 27%, and when run on clusters using the methodology in Section 3.2, this improvement jumps to 30 to 40%. We are able to solve problems of several thousand requests in under five minutes for weekends and under twenty minutes for weekdays, and these running times could be improved even further with parallelization. In the next section, we will discuss how this tractability allows for adaptation of the algorithm for re-optimization in the face of uncertainty.

5.3. Tractable Re-Optimization

A major source of uncertainty in The RIDE's operations comes from cancellations. On average, about 20% of requests are cancelled every day. These cancellations pose a particular issue when they occur after the plan has already been computed and passengers have been promised their expected pickup times, locking in the requests and reducing operating flexibility. Fortunately, the majority of cancellations still occur early enough to allow paratransit agencies to react.

A simple strategy for reacting to cancellations might be to perform the itineraries as planned, skipping any cancelled requests along the way. Although this strategy is easy to compute and execute, it may lead to substantial vehicle empty hours, which will still be paid for by the paratransit agency. Clearly, re-optimization can help improve operating efficiency and thereby lower costs.

In our re-optimization procedure, we shrink the pickup time windows to be within five minutes before or after the promised pickup times, while also respecting the original time windows. We then rerun our algorithm from scratch on this updated request data, excluding the cancelled requests.

Table 6 Running times for TCP (6) on different clustered trip instances from one week's worth of real data, as well as an insertion heuristic. "Cluster BY-BY" refers to the TCP (6) run on the trips generated according to Section 3.2. "Cluster S-BY" refers to the TCP (6) run on the trips generated using Santi et al. (2014). The insertion heuristic is similar to that implemented by Marković et al. (2015). The "Cluster" column is reproduced from Table 3. The "Connect" column refers to solving the TCP. The "Total" column is the total running time. For the two clustering methods, this total includes extra computation time for data processing.

Date	**Requests** N	**Method**	**Running Time (min)**		
			Cluster	Connect	Total
Mon, Sep 10	6,600	Cluster BY-BY	9.6	**2.4**	15.5
		Cluster S-BY	**3.5**	5.0	**13.5**
		Insertion heuristic			23.9
Tue, Sep 11	6,783	Cluster BY-BY	10.1	**2.6**	16.8
		Cluster S-BY	**3.4**	4.7	**13.2**
		Insertion heuristic			25.3
Wed, Sep 12	7,437	Cluster BY-BY	11.7	**2.0**	18.5
		Cluster S-BY	**4.3**	5.0	**16.0**
		Insertion heuristic			30.0
Thu, Sep 13	7,408	Cluster BY-BY	11.5	**2.4**	18.2
		Cluster S-BY	**4.2**	5.0	**15.7**
		Insertion heuristic			30.6
Fri, Sep 14	7,026	Cluster BY-BY	10.4	**1.4**	15.8
		Cluster S-BY	**3.7**	5.0	**14.3**
		Insertion heuristic			28.0
Sat, Sep 15	3,494	Cluster BY-BY	2.5	**0.6**	4.8
		Cluster S-BY	**0.6**	0.8	**3.7**
		Insertion heuristic			6.2
Sun, Sep 16	3,096	Cluster BY-BY	2.2	**0.3**	3.5
		Cluster S-BY	**0.6**	0.3	**2.1**
		Insertion heuristic			5.8

Admittedly, there might be other constraints to consider in re-optimization, such as if certain drivers have been already promised a number of working hours. We omit such considerations from our stylized example and assume that the operator is adaptable.

Our standard metrics of productivity and outsourced requests are reported in Table 7 for both the skipping and re-optimization strategies. In addition, we report the fraction of time between first pickup and last dropoff that cars spend empty. Comparing the values in Table 7 to those for the problem without cancellations in Table 5, we see that the 20% cancellation rate leads to a corresponding 20% drop in productivity for the skipping strategy. Although the shrunken pickup time windows prevent re-optimization from fully achieving the original productivity metrics, we are still able to cut the losses in productivity from 20% to about 5%. These improved productivity metrics are achieved while also outsourcing about 24% fewer requests than in the skipping strategy.

Both metrics reflect that re-optimization makes more efficient use of the available resources, with vehicles spending about 7% less time empty.

Table 7 Performance of re-optimization as compared to a simple procedure that skips cancelled requests.

Date	**Reqs.** N	**Productivity**		**Outsourced**		**Frac. Empty**	
		Skip	Re-Opt	Skip	Re-Opt	Skip	Re-Opt
Mon, Sep 10	6,600	1.62	1.93	420	309	0.194	0.121
Tue, Sep 11	6,783	1.63	1.92	343	305	0.186	0.116
Wed, Sep 12	7,437	1.67	1.96	480	343	0.194	0.121
Thu, Sep 13	7,408	1.65	1.94	433	315	0.193	0.118
Fri, Sep 14	7,026	1.66	1.95	443	333	0.201	0.125
Sat, Sep 15	3,494	1.58	1.88	231	178	0.247	0.163
Sun, Sep 16	3,096	1.62	1.94	152	122	0.222	0.146

Our results illustrate that the benefit of a tractable routing algorithm lies not solely in advance planning, but also enables significant mitigation of losses from uncertain disruptions. We examined a stylized case to illustrate the effectiveness of re-optimization in reacting to request cancellations, and believe re-optimization will prove useful in other situations such as unexpected traffic or vehicle breakdown.

6. Conclusion

In this paper, we have presented a tractable algorithm for solving large-scale dial-a-ride problems for paratransit. Our approach follows the framework of clustering requests, solving routing problems on these clusters to produce a set of trips, and then connecting trips into driver itineraries while respecting labor constraints. The goal was to produce high-quality solutions in reasonable running times for daily planning. To this end, we described a clustering procedure that performs local improvements that are guided by insight from optimization, and formulated a trip connection problem that generated hundreds of driver itineraries in minutes. Our optimization models also allow for outsourcing requests to external TNCs.

On real-world datasets from Boston, we were able to produce solutions in under twenty minutes for problem sizes of several thousand requests, and we report improvements in productivity of over 30% relative to an insertions-based algorithm that is used in practice. The tractability of our algorithm also makes it suitable for re-optimization in the face of uncertainty, and we demonstrate its efficacy on a routing problem with cancellations. With this work, we aim to show that stronger algorithms will go a long way to making paratransit affordable, and that essential government services need not come at inordinate cost.

Acknowledgments

The authors thank Diogo Lousa from The RIDE in Boston for productive discussions.

References

Agatz, Niels, Alan Erera, Martin Savelsbergh, Xing Wang. 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* **223**(2) 295–303.

Alonso-Mora, Javier, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences* **114**(3) 462–467.

American Public Transit Association. 2019. Transit and TNC partnerships. URL `https://www.apta.com/research-technical-resources/mobility-innovation-hub/transit-and-tnc-partnerships/`. Accessed: 2019-05-13.

Baldacci, Roberto, Vittorio Maniezzo, Aristide Mingozzi. 2004. An exact method for the car pooling problem based on Lagrangean column generation. *Operations Research* **52**(3) 422–439.

Baldacci, Roberto, Aristide Mingozzi, Roberto Roberti. 2012. Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints. *European Journal of Operational Research* **218**(1) 1–6.

Bertsimas, Dimitris, Allison Chang, Velibor V Mišić, Nishanth Mundru. 2019a. The airlift planning problem. *Transportation Science* **53**(3) 773–795.

Bertsimas, Dimitris, Patrick Jaillet, Sébastien Martin. 2019b. Online vehicle routing: The edge of optimization in large-scale applications. *Operations Research* **67**(1) 143–162.

Bezanson, Jeff, Alan Edelman, Stefan Karpinski, Viral B. Shah. 2014. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607* .

Binette, Joanne, Kerri Vasold. 2018. Home and community preferences: A national survey of adults age 18-plus. *Washington: AARP Research* .

Borndörfer, Ralf, Martin Grötschel, Fridolin Klostermeier, Christian Küttner. 1999. Telebus berlin: Vehicle scheduling in a dial-a-ride system. *Computer-Aided Transit Scheduling*. Springer, 391–422.

Citizens Budget Commission. 2016. Access-a-Ride: Ways to Do the Right Thing More Efficiently. URL `https://cbcny.org/sites/default/files/media/files/ACCESS-A-RIDE_0.pdf`. Accessed: 2019-05-13.

Cordeau, Jean-François, Gilbert Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* **37**(6) 579–594.

Cordeau, Jean-François, Gilbert Laporte. 2007. The dial-a-ride problem: models and algorithms. *Annals of operations research* **153**(1) 29–46.

Cortés, Cristián E, Martín Matamala, Claudio Contardo. 2010. The pickup and delivery problem with transfers: Formulation and a branch-and-cut solution method. *European Journal of Operational Research* **200**(3) 711–724.

Diana, Marco, Maged M Dessouky. 2004. A new regret insertion heuristic for solving large-scale dial-a-ride problems with time windows. *Transportation Research Part B: Methodological* **38**(6) 539–557.

Fitzsimmons, Emma G., Rebecca Liebson. 2019. M.T.A. Pledges Five Billion Dollars for Subway Elevators. Guess How Many. URL `https://www.nytimes.com/2019/10/07/nyregion/mta-nyc-subway-elevators.html`.

Google OR-Tools. 2019. Google OR-Tools. URL `https://developers.google.com/optimization/routing`.

Gurobi Optimization, Inc. 2016. Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

Herbawi, Wesam, Michael Weber. 2011. Evolutionary multiobjective route planning in dynamic multi-hop ridesharing. *European conference on evolutionary computation in combinatorial optimization*. Springer, 84–95.

Ioachim, Irina, Jacques Desrosiers, Yvan Dumas, Marius M Solomon, Daniel Villeneuve. 1995. A request clustering algorithm for door-to-door handicapped transportation. *Transportation science* **29**(1) 63–78.

Jain, Siddhartha, Pascal Van Hentenryck. 2011. Large neighborhood search for dial-a-ride problems. *International Conference on Principles and Practice of Constraint Programming*. Springer, 400–413.

Lubin, Miles, Iain Dunning. 2015. Computing in operations research using julia. *INFORMS Journal on Computing* **27**(2) 238–248.

Marković, Nikola, Rahul Nair, Paul Schonfeld, Elise Miller-Hooks, Matthew Mohebbi. 2015. Optimizing dial-a-ride services in maryland: benefits of computerized routing and scheduling. *Transportation Research Part C: Emerging Technologies* **55** 156–165.

Massachusetts Bay Transportation Authority. 2015. Net subsidy by mode: Part II. URL `https://cdn.mbta.com/uploadedfiles/About_the_T/Board_Meetings/NetSubsidybyModePartII11182015.pdf`. Accessed: 2019-05-13.

Masson, Renaud, Fabien Lehuédé, Olivier Péton. 2014. The dial-a-ride problem with transfers. *Computers & Operations Research* **41** 12–23.

Mather, Mark, Linda A. Jacobsen, Kevin M. Pollard. 2015. Population Bulletin: Aging in the United States. *Population Reference Bureau* .

Parragh, Sophie N, Jean-François Cordeau, Karl F Doerner, Richard F Hartl. 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR spectrum* **34**(3) 593–633.

Ropke, Stefan, Jean-François Cordeau, Gilbert Laporte. 2007. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks: An International Journal* **49**(4) 258–272.

Santi, Paolo, Giovanni Resta, Michael Szell, Stanislav Sobolevsky, Steven H Strogatz, Carlo Ratti. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proceedings of the National Academy of Sciences* **111**(37) 13290–13294.

Savelsbergh, Martin, Marc Sol. 1998. Drive: Dynamic routing of independent vehicles. *Operations Research* **46**(4) 474–490.

Siddiqui, Faiz. 2018. Wheelchair-accessible Uber service comes to D.C. and five other cities, expanding mobility options for people with disabilities URL `https://www.washingtonpost.com/transportation/2018/11/20/uber-launches-wheelchair-accessible-service-dc-five-other-cities/`.

Sullivan, Gail. 2014. Uber sued for allegedly refusing rides to the blind and putting a dog in the trunk URL `https://www.washingtonpost.com/news/morning-mix/wp/2014/09/10/uber-sued-for-allegedly-refusing-rides-to-the-blind-and-putting-a-dog-in-the-trunk/`.